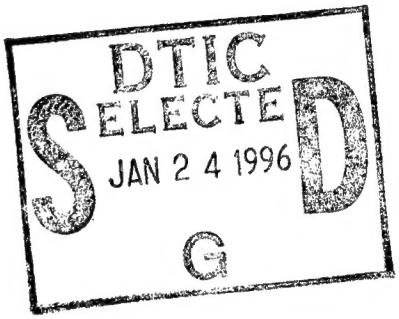
RL-TR-95-209, Vol I (of two)
Final Technical Report
October 1995

PARALLEL SOFTWARE ENGINEERING TECHNOLOGY FORECAST, Blue Ribbon Panel Conclusions

RCI, Ltd.

Carl Murphy



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

RL-TR-95-209, Vol I (of two), has been reviewed and is approved for publication.

APPROVED: Joseph P Cavano

JOSEPH P. CAVANO
Project Engineer

FOR THE COMMANDER: John A Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/ (C3CB), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1995	Final Jul 94 - Jun 95

4. TITLE AND SUBTITLE
PARALLEL SOFTWARE ENGINEERING TECHNOLOGY FORECAST,
Blue Ribbon Panel Conclusions

5. FUNDING NUMBERS
C - F30602-94-C-0108
PE - 62702F
PR - 5581
TA - 20
WU - PH

6. AUTHOR(S)
Carl Murphy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
RCI, Ltd.
1301 East 79th Street, Suite 200
Minneapolis MN 55425

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory/C3CB
525 Brooks Rd
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-95-209, Vol I (of two)

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Joseph P. Cavano/C3CB/(315) 330-4063

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Rome Laboratory developed a "Parallel Software Technology Forecast" to identify the parallel software engineering technology that will be required to meet Air Force needs for mission-critical software in a High Performance Computing environment for the next decade. It concentrated on the quality and cost issues of software development for Command, Control, and Communications (C3I) applications and addressed the following goals: (1) anticipate technology directions of the parallel computer industry and forecast parallel software technology capabilities; (2) identify key C3I factors in the Air Force and show what the implications of HPC might be on the Air Force's ability to develop productive and efficient C3I applications software; and (3) compare and contrast Air Force needs for parallel software technology to that in the commercial sector.

Rome Laboratory assembled a distinguished Blue Ribbon Panel consisting of seven technical experts, and solicited position papers from a broader range of position makers from academia, government, and industry.

14. SUBJECT TERMS
Parallel software engineering, Parallel processing,
Parallel software development

15. NUMBER OF PAGES
28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

C^3I Parallel Software Engineering Technology Forecast
Blue Ribbon Forecast Panel Conclusions

1. PREFACE

Rome Laboratory developed this technology forecast to identify the parallel software engineering technology required to meet Air Force mission-critical needs in High Performance Computing (HPC) environments over the next decade. The forecast concentrates on the quality and cost issues of the software development process for Command, Control and Communications applications (C^3I) as well as performance. The "Parallel Software Technology Forecast" addresses three goals:

1) Anticipate technology directions of the parallel computer industry and forecast parallel software technology capabilities, ie, current trends in software technology related to emerging trends in HPC.

2) Identify key C^3I factors in the Air Force mission and show what the implications of HPC are on the ability of the Air Force to develop C^3I application software productively and efficiently.

3) Compare and contrast Air Force needs for parallel software technology to the commercial sector and identify opportunities effective bi-directional technology transfer.

The technology forecast involved five stages:

1) Rome Laboratory produced a summary version of its "Parallel Software Engineering Technology Assessment" report.

2) Rome Laboratory assembled a distinguished Blue Ribbon Panel consisting of technical experts from industry and academia. The Panel included: Dr Walter Beam (noted author on C^3I and system engineering), Mr C. Gordon Bell, (Chairman, University Video Communications), Prof. Tom Cheatham (Harvard University), Dr George Lindamood (Director of Information Services for the State of Washington), and Dr Jeffrey Mohr (Chief Scientist, Information Technology Solutions).

3) Rome Laboratory announced a Technology Forecast Forum to solicit position papers. A broad range of *position makers* from academia, government and industry responded. Twenty positions were selected for review by the Blue Ribbon Panel as representative of the state of the art. In addition, Panel Members also provided position statements.

4) Rome Laboratory sponsored an invitation-only, Blue Ribbon Technology Forecast meeting at the Orlando Hyatt on January 23-26, 1995. Panel members and position makers interacted to produce the technology forecast. Presentations of each participant's view were followed by separate sessions on C^3I, commercial off the shelf, and parallel technology impacts. The group then collaborated on notes that represent the combined trends, vision and strategy for C^3I parallel software engineering.

5) After the interaction, this author set down the conclusions of the panel and distributed it for confirmation by the Panel Members. "Volume I: C^3I Parallel Software Engineering Technology Forecast -- Conclusions" is the result. Volume II provides an assessment, analysis and confirmation of the Blue Ribbon Panel's conclusions. The title of Volume II is "C^3I Parallel Software Engineering Technology Forecast -- Assessment, Trends, Vision and Strategy." The Volume II Appendices provide an assessment of the technology prepared by Rome Laboratory and brief summaries of the position maker's papers and presentations.

2. EXECUTIVE SUMMARY

Parallel software engineering can fulfill important roles in C^3I systems. Using it, software engineers can overcome the time delay shortfalls of sequential computing. They can provide the C^3I designs that assure delivery of required mission function in a timely fashion. In spite of few examples, the panel believes that parallel software engineers can solve the difficult and complex applications typical of C^3I systems. A parallel capability allows designers to invent advanced warfighter information system applications. However, parallel software engineers need a better process and tool foundation to cost-effectively gain these capabilities. The panel members defined the trends, vision and strategy to define the process. (Note that the panel limited its discussion by excluding signal and image processing, believing that using parallel computers for such structured applications is a relatively well understood process.)

Throughout this report the following C^3I Parallel Software Engineering terms apply:

Mission Function or *Tough Character Applications* -- these combine a mix of high complexity, unstructured (state and data dependent) paths, high dynamic resource fluctuation and frequent synchronization characteristics found in C^3I environments. They are the most challenging computer projects.

Assured Delivery or *High-Integrity-Value* -- these factors are fault tolerance, availability, graceful degradation, security, reliability, survivability, repairability, usability, twenty-four-hour, seven-day operation time frame and the many other factors that are required for the mission in battle environments.

Mission Performance and *High-Time-Value* -- C^3I applications operate where time responses are critical to mission success. Time is the essence of C^3I systems. High-Time-Value can range from hard real-time to use of short time goals that might allow on-line plan changes during battle to increase mission effectiveness.

Parallel Software Engineering -- C^3I systems must be engineered and are typically built by large groups. C^3I parallel software engineering is the collection of processes and methods used to achieve predictable performance, programmability, portability, maintainability, adaptability and low life cycle costs.

2.1 PANEL CONCLUSIONS

The panel set some clear directions toward parallel software engineering processes and infrastructure. The issues of high-time-value, tough character applications and C^3I mission factors express the challenges of parallel software engineering. All must be delivered at once, along with performance prediction at system definition, design and implementation layers.

The panel set the architectural target as a mix of high bandwidth capability distributed nodes and low latency multiprocessors. This means that parallel software engineering tools can separately address applications suitable for message passing, applications requiring low latency, and the tools needed to interface and control the interaction between them. Research should focus on performance prediction and effectiveness and use of parallel software engineering as a tool to achieve C^3I mission factors while meeting mission time response needs.

The panel also laid out parallel software engineering goals that affect the way C^3I systems would be engineered. The justification decision boundary between choices of commodity priced COTS, modular board level COTS, and military-unique approaches should be investigated and understood. Incremental building methods should allow incremental build and test methods. Probes and information gathering methods are indicated to take advantage of the "daily" exercised and evaluated C^3I system. Engineering refinement layers are needed to facilitate change and new mission needs.

To meet these goals, parallel software engineering researchers will have to define new relationships with COTS computer vendors and integrators to determine the boundary of COTS suitability. We will also have to build relationships with COTS system vendors to establish testbed demonstrations. Also, different relationships are needed with operating C^3I organizations to frequently communicate requirements, operations, test results and effectiveness. Meeting these new challenges requires merging parallel software engineering research to meet a wider range of commercial and military needs.

2.2 ASSESSMENT

C^3I software engineering encounters the most difficult development problems known in computing. Mission constraints preclude use of over-specialized research computers and software methods.

Today's parallel technology performs well for well-structured, high-throughput applications. Those machines concentrate on throughput and structured applications, instead of general parallel computing, making system building and life-cycle support expensive and risky. This is because C^3I applications are complex, dynamic, and closely coupled to other applications. They are also constrained by stringent time response and mission integrity factors. As a result, our parallel technology provides an inadequate foundation for software engineering of complex and dynamic portions of C^3I systems.

Because of this specialization, applications do not scale. Hardware and operating software do not give the assured delivery (high-integrity-value) and fast time response (high-time-value) required in C^3I missions. The C^3I industry needs a more general purpose parallel capability that can be programmed for fast time response and assured delivery while executing complex and dynamic mission requirements.

Predictability of time response, integrity, and project management metrics are also necessary. Program managers expect that parallel software engineers deliver predictable time response in C^3I systems. They expect these engineers to meet C^3I mission factors for assured delivery in a well planned, organized development process. Parallel software engineer researchers should provide the process and tool foundation for C^3I builders to do their job.

2.3 TRENDS

Demands on parallel software engineering for C^3I systems will increase. This is due to a shift to joint task force theater operations and the need to put the warfighter in the loop. COTS capabilities will continue to grow by a factor of four every third year.

The panel reasonably expects increased demands for high-time-value, tough character applications and difficult C^3I mission factors. Parallel software engineering should aid in providing time constrained solutions to C^3I mission factors. These factors are the following: reliability, availability, security, fault tolerance and graceful degradation. C^3I's high-time-value and tough character applications render the message passing techniques of scientific research parallel methods ineffective. Shorter time scales, dynamic reconfiguration, complex data interaction, visualization demands, and collaborative interaction will grow in importance. Interfaces and information interchange will become a critical use of parallel computers in C^3I systems.

The panel projected a five year trend for commercial parallel computing systems. It includes the following:

- nodes with a 1 GMIPS Pentium-like microprocessor
- 256 to 512 MB memory
- a 20 GB disk drive
- Windows NT operating system
- Client programming model using "visual" programming language

- ATM-based distributed network
- data and multimedia access by SQL

The programming model is that a visual language with SQL extensions will deliver graphical user interfaces to obtain data from the network. Commercial systems might adopt object-based technology as its common data access means in order to access multimedia or complex data structures.

The vision includes only very high volume hardware, data base and operating system software and application package software. In this model the client desktop process views the network as a massive server. The system behind the network connection is featureless as far as the user programmer is concerned. Expected technical capability increases for single microprocessors, memory, disk units, and network bandwidth capacity allow the conjecture that almost all computation would be done by the client desktop workstation without any large scale application servers. A totally distributed peer-to-peer system results. (An exception would be interfaces to any pre-existent legacy systems.)

The resulting commercial distributed computing system would be suitable for many C^3I application needs. Throughput applications would use ATM's high bandwidth (655 mbps). The distributed system would be effective for almost all of today's successful parallel applications -- transaction and decision support business applications, replicated engineering applications, and message passing scientific research.

However, fast time response and assured delivery of function require that some tightly coupled computing remain. Competitive forces for strategic decision making and concurrent engineering will create the need for the capability for high-time-value and high-integrity-value solutions. This yields an architectural separation with a distributed COTS foundation for some C^3I parts and some high-time-value parallel computers for others. This dichotomy separates bandwidth-only throughput from low-latency fast time response demanding applications.

The client/server commercial programming model is good because it does not allow the application programmer any detailed knowledge of the system structure. Presently, SQL database management systems provide this model for data retrieval. However, the client/server model lacks a parallel application language, thus requiring that all processing past data retrieval be executed locally. There is no widely accepted client/server model for users to express execution on network or parallel application servers without explicit expression based on the system's structure.

2.4 VISION

C^3I demands, budget constraints and COTS trends require improved parallel software engineering approaches. These demands mean that parallel software engineers must find better development cycle processes that incorporate high-end COTS and parallel machines. Understanding the justification line between uses of COTS, software engineered supplements and military-unique computing is a critical skill.

C^3I systems have more stringent needs than do commercial systems. Their time cycle is also much shorter. In addition, due to battle engagement use, a computing error has greater potential for causing loss of human life. To meet these stringent demands, a C^3I software design often must include the following:

- meeting time goals (high-time-value) coupled with C^3I mission factors (high-integrity-value)
- diversity of applications
- distributed data repositories
- multiple access mechanisms
- dynamic resource allocation

- quick visual and imagery processing
- predictable performance and integrity
- tough character (complex, dynamic and closely coupled) applications

In the future, C^3I software engineers would use tightly coupled parallel computers to meet those needs.

However, the panel expects that throughput and well-structured codes would, instead, use a COTS network of nodes with the client/server programming model. For C^3I use, the network would use very high-end (speed and reliability) nodes and ATM switches. Such a foundation could meet network pathway bandwidth and single workstation application needs of many throughput and highly structured applications. However, it might not meet all C^3I demands, especially where high time value and high integrity are necessary. Potential shortfalls of using an exact copy of the COTS are the following:

- anticipated ineffectiveness of SQL for diverse C^3I data
- possibility of a short fall in ATM latency or bandwidth capacity for tight synchronization and data transfer respectively
- weakness of object-based methods for very large project design efficiency
- performance inadequacy of Windows NT to provide real time capability

The conclusion is that this "likely" COTS foundation needs a supplement of tightly-coupled, parallel computers with corresponding parallel software tools to help deal with the resulting complexity.

Software engineers would target their work toward a high performance COTS network. However, to meet well-defined mission time constraints, assured delivery and difficult "tough-character" applications, they must supplement it by adding tightly coupled parallel processors.

2.5 STRATEGY

There will be no silver bullet that solves these challenges to C^3I *parallel software engineering. However, the panel outlined a path toward gaining the necessary foundation and knowledge to do timely and responsive* C^3I *engineering.*

The system designer should base designs on an architecture similar to the COTS fabric and must justify any parallel supplement. The panel set the following goals for parallel software engineering research:

1) Generate C^3I system level design tools that:

- model the COTS network and pockets of tightly coupled parallelism dichotomy,
- predict performance (time response) and integrity (assured delivery)
- establish system engineering metrics
- define the justification line between use of parallel and the COTS distributed fabric

2) Develop interface mechanisms for:

- connecting parallel with a COTS network foundation architecture into C^3I systems
- achieving C^3I mission factor capabilities
- testing and monitoring C^3I exercises to capture mission factor and time value information

3) Reduce the cost and effort for development and life cycle support by:

- finding architectural independent design and programming models
- defining architectural independence as generational portability of today's solutions to tomorrow's computers

- insisting on refinement and performance prediction at each level (system, design and programming)
- adding the necessary coordination languages, effective protocols, and hardware interface drivers to demonstrate system level middleware functions

4) Find methods for:

- achieving dynamic reallocation of resources for meeting mission needs
- responding to collaborative interaction by multiple users
- inserting test or monitor probes for performance modeling and recursion testing
-

5) Experiment with COTS technology to enable continuous tracking of key commercial technologies, including:

    * operating systems
    * languages
    * SQL
    * object techniques
    * visual programming
    * ATM data transfer

- develop a demonstration testbed for emulating COTS architectures
- cooperate and communicate with the COTS industry to define and continuously revise the COTS network of nodes coupled with tightly coupled multiprocessors

3. BACKGROUND

C^3I parallel software engineers should focus on the most difficult character applications coupled with the high-integrity and high-time value that define C^3I computing. Parallel software engineers must understand COTS technology so they can use its rapid technology gains wherever appropriate.

The Blue Ribbon Forecast Panel for Parallel Software Engineering Technology Forecast convened to obtain the expert opinions and forecasts of leaders in C^3I related parallel technology. The panel included C^3I experts, commercial, government, and academic members. The panel organizers selected them due to their cognizance of architectures, applications, components and markets for parallel systems. In order to provide the panel with a diverse set of ideas on parallel software engineering, the organizers assembled position makers to obtain their ideas and projection of the future of parallel software engineering. Position makers represent a wide range of activities, among which were the following:

- industry level -- machine cost models, research center activities, industry health status and future industry viewpoints
- system level -- system engineering, engineering applications, C^3I systems, and dynamic control
- programming models -- virtual machines and automated parallel programming
- programming tools -- visual, components, libraries, and debugging
- system effectiveness -- domain specific objects, libraries, and hardware architecture target development tools

The Blue Ribbon Forecast experts focused their attention on C^3I system requirements for:

- high-time-value applications in C^3I systems
- high-integrity-value C^3I mission factors (fault tolerance, security, graceful degradation, physical constraints, etc.)
- difficult "tough-character" applications (dynamic resource demand, symbolic interaction, frequent synchronization, and extensive complexity)
- software engineering issues necessary for C^3I-like projects: programmability, rapid development, and maintenance ease combined with effective performance and portability with predictable design and development performance allocation

4. SCOPE, DEFINITION & ASSUMPTIONS

The decision and information delivery system requirements of C^3I *systems are challenging.* C^3I *Parallel software engineers should focus on warfighting planning, information generation and command decision making parts of the system. Budget constraints will require maximum use of COTS hardware, software and rapid development processes. Commercial information system COTS technology also poses a threat to our forces due to its widespread availability to others. It also raised expectations from commanders who demand rapid delivery of capabilities they might see in computer stores.*

The panel set boundaries on the technology and forecast elements that it was to consider. The panel believed that the opportunities of parallel computing should derive from the mission needs, not from zealous enthusiasm over a particular technology. However, imposing this discipline should not inhibit the invention of new (or improvement of old) C^3I applications and capability. The panel stressed the importance of the potential of parallel technology to meet high-time-value, high-integrity-value C^3I component needs.

4.1 COMMERCIAL-OFF-THE-SHELF BOUNDARIES

The panel considered that one of its important roles was to determine the directions and condition of the commercial-off-the-shelf (COTS) parallel industry. The forces of change in commercial technology are the driving forces in computing. Also the economics of building military systems require maximizing use of commercial technology to reduce acquisition, development, and life time support costs. The reasonably expected state of commercial enterprise will support some C^3I computing needs. However, its use will require aggressive study of the boundary between COTS, parallel machines, and existing C^3I systems. Designers should use specialized military or parallel technology only when necessary to meet mission needs. Parallel software engineers need to define the right strategy for incorporating COTS for optimizing C^3I capabilities. The reasons are that COTS components have rapid technology improvement cycles, broad acceptance and short learning curves.

4.1.1 Rising expectations from COTS technology

The performance and wide spread availability of COTS (workstations, networks, operating systems, data and object bases, languages, visualization, virtual reality, voice and pen interaction, and CD-ROM storage) will raise expectations on C^3I systems. Parallel computing for business, engineering, and other government processes includes a variety of architectures. These typically include a number of small symmetric multiprocessors, some clusters of networked computing nodes and a few tightly coupled multiprocessors. Tightly coupled multiprocessors will be market viable only for high-time-value applications.

4.1.2 Potential threat from COTS technology

Third world countries might use COTS equipment to significantly increase their command and control and information warfare capability. Such a threat requires that our C^3I builders fully understand the capabilities

of COTS equipment and its potential for command and control applications. Our challenge is to remain far ahead of the COTS capability through innovation and skillful application of parallel computers within a COTS-based C^3I fabric.

4.1.3 C^3I Boundaries

To concentrate on the right issues, the panel defined the boundaries of C^3I systems as the following functions necessary to conduct Air Force warfare:

- data and information fusion
- information generation, display and visualization
- decision making
- human interaction with the process of command, control and intelligence
- action and reaction control
- warfare planning, direction, evaluation, and awareness
- managing the wide range of information that will converge on command centers

Therefore, C^3I systems are planning, information and decision making elements of Air Force Warfare. The panel excluded certain subsystems, eg, sensor and image data sources for the C^3I system and the external weapons and delivery systems that take commands to respond to a threat or meet mission goals. (We define these as effectors.)

4.1.4 Parallel software engineering boundaries

The directions of parallel software engineering are dependent upon sequential software engineering. Other research technology developments may provide different supporting contributions for C^3I systems. Therefore, the panel attempted to observe duplications of parallel software engineering to avoid overlap with other research programs. These boundaries are often difficult to establish. Several government programs already perform research and development in the following: operating systems, object design, scheduling, application programming interfaces, heterogeneous programming, application scheduling, scientific collaboration and distributed applications. Likewise commercial development continues for distributed objects, new operating systems, visualization, multimedia, virtual reality, data base access, protocols for Asynchronous Transfer Mode (ATM), and languages for more effective development of data and object manipulation. A research plan should avoid duplication. However, parallel software engineering researchers must establish a relationship with other research programs and C^3I providers. Historically, other research programs have concentrated on highly structured scientific research applications that are not effective for C^3I parallel software engineering. An interface to expose the C^3I software engineering needs to other major research efforts is, therefore, necessary.

5. ASSESSMENT OF C^3I REQUIREMENTS & DEMANDS

C^3I systems will often need to use parallel processing to meet response time, physical constraints, and new mission functions. Because parallel computing improves response time, new mission capabilities are feasible. Designers use this approach to radically change mission requirements. They can radically improve a C^3I system's war fighting capability by application of parallel technology.

The character of C^3I applications is very different from scientific research, business transaction processing and other well-known parallel applications. Those character differences are a combination of complexity, interaction rate, and resource demand. Extensive (unstructured), frequent or symbolic (logical and decision) interaction, and dynamic resource demands are the most demanding. The panel referred to these as the

"tough" applications. The "standard" parallel applications typically have a structured, (large) message passing, infrequent interaction and static resource character. The greatest challenge to applying parallel computers to C^3I systems is in dealing with tough applications with coupled high-time-value and high-integrity-value.

5.1 Continued Requirements for C^3I Mission Needs -- Tough Character Applications, High Integrity and Time Value

5.1.1 Assessment of Parallel Technology and Software Engineering

Present COTS parallel computers have not provided an effective capability for delivering C^3I *mission factors such as reliability, availability, security, fault tolerance, and graceful degradation.* C^3I *mission factors will continue to be important discriminators between* C^3I *and COTS parallel systems. Software engineering tools that give better* C^3I *factors would have a high value in commercial systems.*

The panel viewpoint taken is that C^3I system should have hardware components of commodity COTS, specialized (possibly parallel) COTS, and militarized components. These components would be workstations, networks, specialized processors, operating systems and runtime software for implementing C^3I factors and meeting tough character applications. A future view of C^3I systems is one based on a high level of available COTS computing machinery and software. The panel raised some concerns about over-enthusiastic COTS adoption. They also raised concern over potential long-range disadvantages of COTS. One example is the frequent and sudden obsolescence, common in commercial markets. The panel reasonably expects significant cost advantages through COTS. The panel recommends setting the goals of parallel software engineering by reflecting on the future of COTS components.

5.1.2 C^3I component development

Until parallel software engineering environments are available that can simulate the entire C^3I system, designers will use allocation of time and function to components. Software engineering is necessary due to the large scope and difficulty of building C^3I systems. The necessity to use system and software engineering methods to build C^3I systems is due to their size, high-time-value, high-integrity-value, and tough character. Since C^3I design and development require many people, the visibility of the design and its implementation are very important.

Parallel computers exacerbate the situation because their performance (time to completion) varies widely when the design mismatches the architecture. Designers allocate function and performance to components when designing a C^3I system. This is difficult even when the underlying capabilities of the computing resources are consistent and steady providers of computing capacity. Without the capability to predict the behavior of a parallel element in the system, the design risk becomes excessive.

Designers should not use parallel computers unless they identify clear advantages. In other words, they should not use technology for its own sake. However, when applications honestly require performance and integrity levels only obtainable through parallel computers, then system designers must be able to predictively use parallel technology. Parallel engineering researchers should create suitable metrics to aid designers in making these decisions.

Knowledge of dependable and predictable completion times for each component are required to conceptualize and develop architectures that deliver high value time. In addition, the extra computation required to deliver all required C^3I mission factors must be predictable at design time, prior to detailed mapping steps to a specific architecture. Delivering reliable time of completion predictions on diverse character applications is a necessary design tool. Designers also need tools to help in delivering C^3I mission factors and predicting the associated performance penalty. Tools that provide information about the

application's character are also necessary. It is also necessary to develop methods to incorporate C^3I factors such as time guarantees, resiliency and integrity, and physical constraints imposed by operating in aircraft.

5.1.3 Complex data manipulation

The most compelling feature of C^3I parallel software engineering is that applications are complex in all dimensions -- processing, data location and organization, interaction of components, response to changes, and necessity for integrity in combat situation. In addition, C^3I data fusion techniques will increase in complexity. Additional complexity comes from added data sources, higher precision, and lower false detection rates. In addition, future systems require "information" fusion. This combines today's data fusion results into multiple information sets for use in different applications.

Commercial users will develop data mining techniques that use data content and histories to improve the capabilities of the run time system. Similar adaptive improvement of operational capability using data mining would improve the quality of C^3I systems. These future uses require complicated data manipulation to such a degree that systems will require parallel computers to meet time goals. Single workstation technology advances alone are not adequate to meet these needs.

The multiple sets of applications that comprise a command and control system take widely divergent forms. Each form may have widely varying optimum data organizations. Relational data base managers are most prevalent in commercial systems. Their use in accessing multiple data arrangements is not as effective and is difficult to apply for highly diverse component access methods. An inefficiency in both access time and storage volume results. The C^3I system has highly divergent applications with distributed data repositories that do not match a single "normalized" relational model which is tuned for high throughput transaction processing.

5.1.4 Shorter command time scales

Time is the essence of C^3I *systems. Predictable time response and assured delivery are the essence of* C^3I *system design.*

Every time period in a C^3I system grows shorter when the focus moves from monolithic central control to warfighter information delivery and decision making. Parallel computers most reasonably fit into high-time-value C^3I components. One example is reduction in time for creating air battle command plans for a day's actions. The contribution of parallel performance could enable on-line plan changes-based on intelligence as it arrives during the day, allowing the plan to be adapted to meet immediate mission needs. Such an application would require collaborative action in setting up the daily plan and a rule-based system for safely changing parts of the daily mission during its progress. Users would base changes on the rule base and real time information gained during the day's operations.

Parallel computing is one method where the C^3I systems can receive the steady stream of input, process event detections, make command and control decisions and meet a time response goal. The decision support actions take several streams of events from different sensor streams. It "fuses" them into higher a level set of event information and associates it with other information sources. A higher level of fusion might combine these with information-based on other data types, such as observations, satellite imagery or human input. The system evaluates these information events, often using empirical methods, to identify objects or activity, determine its value or threat and decide upon the proper response or attack. Detection of possible events and determination of a target must have high signal to noise and low false detection performance. In many cases, the system must perform this processing within a very short time frame, otherwise it is useless. Therefore, C^3I mission success commonly depends upon meeting time response goals.

5.1.5 Dynamic reconfiguration

In aircraft space limits physically constrain C^3I system facilities. Therefore, hardware serves multiple purposes. Collaboration among commanders will also become more common. Dynamic reconfiguration is also necessary to achieve graceful degradation and system integrity in combat and information warfare situations.

Parallel computers in C^3I systems must have the capability for dynamic reconfiguration. In this case, components required in different time domains may use common resources. For example, a detected threat might cause redirection of processing. In dynamic reconfiguration, the system determines which of the components to run at any given time. By using the resources to best advantage, designers can have the maximum capability for the minimum hardware costs. Often computer resource demands come from new events or from computed information. This requires that the system to respond and use its machinery differently. This dynamic change capability is not typical in today's common parallel architectures. Often an event identification perturbs processing control. Also a human can interact unexpectedly, basing their decision on displays and visual processing. When this happens, the mission may change. The system must switch all its resources to the highest priority and may require that the computation change significantly to perform a new function.

Parallel computation to accomplish these fixed time response computations must have the same capability. Dynamic resource allocation is a necessity for meeting C^3I mission needs. Of growing importance is the capability for multiple users to interact with a computation as it progresses. Collaborative techniques will eventually contribute to concurrent engineering or other industrial design and information decision making. This technology might also impact the designer's expectations of a C^3I system. Collaborative interaction will be an important factor in increasing demands for more rapid refinement of decisions through multiple scenarios. Parallel computers can solve these problem types.

5.1.6 Visual and Imagery impact

Humans have limited capacity to absorb information. However, their ability to analyze a multidimensional situation picture is far beyond computer capability. Getting that picture and supporting information to decision makers is now feasible. This places a new demand on the C^3I system to create and present the right information and pictures for full situation awareness without overload. The C^3I system of the future will move away from simple displays of alphanumerics and line graphics with text overlays for information display. C^3I system requirements will follow commercial systems and provide significantly higher information content screens. These include scene images, high resolution vectors, symbolic icons, tables, charts, etc. Providing this extra information may reasonably require video storage, added bandwidth communication, and selection mechanisms for human interaction. In addition, designers will include new forms of image analysis and comparison, possibly directed by human controls. This added capability demonstrates significantly increased expectations of access to information and requires high performance processing. A higher demand on computing capability results.

Visual displays, with much higher information and data content, will drive designer's expectations in future C^3I systems. This may change the organization of processing, transfer and storage of symbolic and real imagery significantly. Both parallel computers and high performance networks may be necessary to perform scene analysis, compression, storage, transmission, etc. Three-dimensional scenes, as that possibly provided in new displays like virtual reality, require ray tracing. C^3I applications will have similar imagery for radar scenes, human observations, events, and tracking. The system designer must provide image processing and symbolic decision processing in collaboration with information fusion and human interaction. This is a type of image processing that falls within the C^3I system, at the information generation and its intelligent use for decision making involvement with the user.

5.1.7 Global interfaces and information interchange

Change, flexibility, less-structured and distribution requires exercising the system for validation and testing. Parallel software engineers must provide interfaces to allow for constant refinement of the system through information collection and operational testing. There is a need for a more closely coupled parallel tool set with actual C^3I system operations. Understanding the application's character, the costs and methods of delivering C^3I mission factors, and the necessity for time guarantees requires information collection and knowledge building that many research parallel tool developers presently lack. The parallel software engineering program needs ways of transferring information on fundamentals about the character, factors and high-time-value of C^3I systems. Software engineering for parallel systems needs to be in-the-loop to enable its progress to bring the full value to C^3I systems. These demands on C^3I system change the importance of interfaces to other systems, typically causing an increased input/output rate.

Parallel computers play an important role as information and data storage and buffering mechanisms for input and output and for transfer of information between C^3I system components. Additionally, the parallel computer may be necessary to transform data so that each C^3I component capability meets its time goals. In addition, the parallel computer makes it feasible for software researchers to create the capability to meet C^3I factors, or for the execution of applications with a tough C^3I mission character.

Advanced methods of decision making incorporate intelligent and adaptive methods. They must constantly improve themselves by rule changes or by training. To accomplish this improvement it is necessary that the system have feedback or information gathering capability to achieve its full potential. In-the-loop system probes and data collection are important components for the development of future C^3I systems. An information-gathering probe would be a valuable tool, especially if it were a part of the development process during architectural concept, design, testing, exercises, and life cycle support. Much of the value is the feedback of information on the adaptation and evolution of the system to achieve improved results over time.

C^3I systems operate within an environment where existing "legacy" equipment already operates, and where external system interfaces play an important role. Just as business systems have the necessity to deal with legacy systems, so do C^3I system components. Due to the long life time of C^3I systems, designers must consider that each parallel component will become a legacy of its own. Designs should be extensible to accommodate mission and technology obsolescence. The model is that parallel computers go into the fabric of existing and future C^3I systems only where designers prove them necessary. The panel reasonably expects this for high-time-value and tough character application needs. Therefore, an important component will be the interface mechanisms between parallel and COTS systems. Common sense dictates that system interfaces need a balance of bandwidth capacity and low latency to match the interfaces for high-time-value cases. Simulation and analysis of the C^3I system can give some insurance for achieving the time goals necessary in high-time-value applications. The panel does not expect that scientific research center methods will be significant aids in solving this issue because they are throughput rather than high-time-value oriented.

6. VISION OF COTS PARALLEL TECHNOLOGY AND SOFTWARE ENGINEERING

Parallel software engineers will adopt high end COTS technology to a greater extent. One vision is for a dominance of desktop computers interconnected by high performance ATM networks. In this view, all application processes have local views of data or images accessed from the network. In this model, the only parallel application necessary is the SQL access of data. The reasonable expectation is that these general COTS architectural directions will many solve C^3I throughput requirements. However, the need for mission factors, solutions to tough character applications and high-time-value of C^3I systems requires additional

pockets of tightly coupled parallel computers to solve these special cases. These may also be COTS supplied, but may require special parallel software engineering methods to achieve the necessary mission. Military-unique problems would also be present within this framework.

6.1 NETWORK OF HIGH PERFORMANCE PERSONAL-COMPUTER NODES

A broad vision of the future COTS parallel industry agrees with Gordon Bell's vision of COTS industry based on his Scalable Networks and Platforms (SNAP) architecture. SNAP has the following components:

- nodes based on the highest performance but largest volume microprocessor (likely to be an Intel-based computer with 1 GMIPS, 128 MB memory and 20 GB disk drive by the year 2000)
- some nodes would have a four-processor commodity level symmetric multiprocessor (The Compaq ProLiant is a present day example.)
- a single, standard, commodity-priced operating system with network and multi-user capability (Windows NT is the top contender -- Unix might become outmoded due to high support costs of its many variants.)
- Asynchronous Transfer Mode (ATM) would be the local, wide area, and global network connection
- all data access would be by SQL queries into the network or disk unit
- multiple applications could operate at each node
- computing would be peer-to-peer instead of client server (the entire fabric of nodes and ATM switches would be a virtual server whose architecture is hidden by the data base manager)
- no shared memory is provided by this model -- to give scalability of the nodes and network
- metrics of the SNAP network are message delay (latency and overhead) and bandwidth limit delay (gap)

The continued processing capability increase by a factor of four every third year, economy of scale and better learning curves are the primary driving factors. The result is that commodity pricing lets almost all applications use the ubiquitous SNAP network of nodes. This architectural model represents the expected trend target of commercial throughput computing.

6.1.1 Pockets of tightly coupled nodes

The bulk of the market would tend toward the commodity market represented by the SNAP network. However, there remains a commercial market for decision support machines for high-time-value and high-integrity-value applications. There will be a growing number of large complex applications that require higher performance. Two routes are open to the designer: distributed message passing and tightly coupled multiprocessor servers. For both routes the SNAP network provides the basic fabric. The SNAP architectural model did not include coordination languages to run concurrent systems over the network, all applications were to be single client-based. Coordination languages let designers set up pipelines or concurrent replicas in a SNAP-like network. Middleware control software is necessary.

The panel also expects that today's message passing applications will fit well on the SNAP architecture. The distributed computing route requires a coordination language for distributing large grain applications across the network. Coordination languages let designers set up pipelines or concurrent operations in a network. This middleware control software is necessary for fitting today's message passing applications on the SNAP architecture. The projected SNAP architectural model does not include a coordination language to run concurrent applications over the network. In the SNAP model, all applications run in a single client node and get data from calls into the network. However, organizations often find significant competitive advantage by using high-time-value applications. Many organizations will continue to seek solutions that

give them competitive advantage. This favors multiprocessor servers. Hardware vendors, with a base of commodity sales to the SNAP-like COTS requirements, would find their highest value added and profit from these servers in pockets of high-time-value or high-integrity-value.

6.2 COTS NETWORK OF NODES -- A FABRIC FOR C^3I SYSTEMS?

The reasonably expected vision of a C^3I system is one that incorporated a maximum amount of COTS elements. However, C^3I designers must provide for high-time-value cases, high integrity C^3I mission factors, and tough character C^3I applications. Commercial client desktops probably would not deliver the combined high-time-value and high-reliability-value required in C^3I systems. However, commercial vendors already have fault tolerant and high performance servers. Substitution of these high-end COTS servers may provide a COTS foundation for meeting high integrity C^3I mission factors and tough character applications. An important tool will let engineers design and build a C^3I system out of a high-end COTS fabric combined with COTS parallel components.

6.2.1 Anticipated Shortfall of COTS Capabilities for C^3I

C^3I systems have shorter time scales, attach more importance to integrity, serve for a twenty-four hour day and are constantly on-line. They should gracefully fail, delivering reduced capability even when components are lost. This capability brings added costs. As a result, the goals and requirements of the COTS-SNAP architecture and those for C^3I systems diverge. A key question is determining if COTS technology is adequate for the C^3I mission. Parallel software engineers will need in-depth and complete knowledge of COTS technology to arrive at the right answer. Engineers must gain the experience necessary by continually developing information collection at these interfaces.

Today's COTS parallel computers are highly capable within very specific markets. These are: scientific research, multi-user replicated engineering, fault-tolerant transaction processing, and batch decision query applications. Projected designs for multimedia, communications switches, and replicated engineering systems suffer a similar narrow specialization. C^3I applications require general purpose parallel computing to meet the needs of unexpected results of data fusion, information fusion, decision making, and control, planning -- all within high-time-value and high-integrity-value constraints. Most of today's COTS examples are useful only for multi-user throughput. In addition, there is a difference in the depth of concern for the high-integrity-value C^3I mission factors such as the following:

- reliability and fault tolerance
- rapid software development, easy maintenance and enhancements
- security
- graceful degradation
- dynamic reallocation
- physical constraints

Some commercial systems provide high availability and fault tolerance, for example, systems for transaction processing. These represent a high-integrity-value COTS market but are specialized to high throughput transaction rates instead of high-time-value decision making.

The necessary component upgrades for using the COTS-SNAP architecture in a C^3I design are critical to understand. C^3I systems would not use commercial communications networks. However, other aspects of the system would be similar. This means that independent vendor applications, software tools and languages, training, and other market infrastructure capabilities would transfer into the C^3I system. Significant cost savings accrue even though hardware components are upgrades from the least expensive commercial equipment. The conclusion is that the COTS fabric provides a model for the C^3I one.

The greatest future challenge may be to solve the toughest of applications needed in C^3I systems. Designers would use pockets of tightly coupled parallel computers for meeting high-time-value, C^3I factors, and tough character missions. A frequent exposure at the interface between the projected SNAP fabric and multiprocessors is necessary to understand that boundary. This mixed COTS fabric with pockets of high-time-value machines is the reasonably anticipated network design. It should be the starting point for military and C^3I applications.

6.2.2 Data and network access

The complexity of C^3I information location and organization may be significantly higher than in commercial systems. Parallel software engineers will have to be aware of and correct COTS weaknesses.

A C^3I system must organize and communicate C^3I mission data and information for rapid access by multiple applications. In addition, the times allowed to access information is much shorter than in commercial systems. There are several areas of concern:

- high-time-value access during dynamic reallocation and changes in application due to threats, plan changes, and defensive responses
- multiple organization of data for different C^3I components within the same system
- access to information must be highly secure, yet available to the applications that require it
- network protocols must provide for guaranteed communications, assurance of resources, and process adaptation to the demand of widely varying event information
- the correctness of the system has human life consequences not found in business systems that will dominate the directions taken by the COTS architectures
- suitability of SQL as an effective data access method in multiple application accesses and within high-time-value constraints
- suitability of ATM latency in high-time-value systems
- reusable methods to efficiently implement C^3I factors and develop software for tough character applications

7. VISION FOR C^3I PARALLEL SOFTWARE ENGINEERING

Use no technology for its own sake.

Software engineers must take total procurement cost conscientiousness, including software development and life cycle considerations. The panel recommended a strategy to maximize the use of a COTS fabric as a foundation. However, parallel software engineers should also use COTS parallel and military-unique computers where they are necessary to meet high-time-value goals, C^3I mission factors and character. Total procurement cost conscientiousness is the metric, not hardware cost alone. This strategy requires programmability combined with performance and portability. It also requires that the underlying system concept have a close relationship to COTS where it applies. An important requirement is to know how to determine the relationship between COTS-SNAP and parallel computers. That interface is one of the challenges a parallel software engineering research program.

7.1 C^3I PARALLEL SOFTWARE ENGINEERING

Parallel software engineers need system engineering tools that allow behavioral simulation and mission performance evaluation during design, development and coding.

A suitable computer aided system design tool is necessary to design the interfaces in C^3I systems. These tools are necessary to allocate function and performance to components, and predicting their performance

and C^3I mission factor capability. A designer typically iterates their design until they meet all mission needs. One result of the panel's vision is the identification of a tool to determine the interface between the SNAP-like architecture COTS and the parallel machine part of the system. As the designer attempts to allocate functions the design tool would predict performance on the SNAP-like COTS architecture. When the designer cannot meet high value time limits the tools should allow introduction of a series of sizes of parallel architectures to meet the high-time-value goal. Thus, the tool lets the designer find the decision surface between SNAP-like COTS fabric and COTS modular parallel or military unique components.

A system design tool provides for an overall system allocation of components. In addition, programmers need application development tools for design and programming of applications. Engineers need methods of application development that include performance prediction and specification execution. Refinement methods, object-oriented methods, specialized components and virtual computing models should be considered and placed intimately into the loop of C^3I exercises, testbeds, and applications.

The panel found that the Bulk Synchronous Parallel (BSP) model eased discussions of architectures and applications. BSP uses super step barriers. It has well-defined terms for communication-to- computation ratio and barrier delay costs. A useful tool is one that would let C^3I system engineers emulate SNAP-like architectures on present day parallel computers. To do this, its developers would have to normalize for projected bandwidth, processing, and storage capability. The BSP super step barriers could be useful in that normalization. If engineering tools could match the architecture to a cost model it could be easier to develop virtual machines as targets for programming tools. Experiments in emulation of predicted COTS systems provide a test bed to allow quantitative performance engineering that attains C^3I mission factors. In addition, designers could explore techniques for testing of tough character applications.

7.2 OBJECT-ORIENTED DESIGN, DEVELOPMENT, AND PROGRAMMING

Object-oriented is the label often used to mean portable, reusable, rapidly developed, and easily maintained programs. However, object-oriented methods are not a silver bullet that makes all progress easy. These methods are often hard to build for effective operation and new designs often require changes to meet all goals. However, if they establish the right foundation, they can build better methods for solving the complex problems posed by C^3I parallel software engineering.

Object-oriented design and development methods have made a significant impact on display screen programming, data exchange between independent applications, and organization of access to data objects. Commercial system and application programmers have accepted these methods. Small programming groups are making significant productivity gains through object reuse, easier software maintenance and efficient enhancements. There are questions about how to overcome performance loss in C++ when compared to compiler optimized C, how to best use C++ in large projects, and how to provide better design visibility. There is also doubt about using object-oriented methods as a cure-all. Others were more optimistic but recognize that a solid foundation is necessary to gain the advantages of object technology. Object-oriented methods are playing an increasing role in commercial systems. This technology has solved the productivity problems for creation of displays. It has reduced the screen building and human interaction programming task significantly. Likewise the language typically chosen by commercial support staff is also becoming object-oriented. System design techniques are also following this path. However, these are not yet mature.

Some of the panel members believed that SQL was not adequate for the commercial systems of the future. They found that programmers would most likely program using a combination of visual object-oriented languages. Examples are Visual Basic or Smalltalk for screen and SQL calls and C++ for higher level system programming.

8. STRATEGY FOR C^3I PARALLEL SOFTWARE ENGINEERING

The vision of the forecast panel for parallel software engineering concentrates on how to make parallel computing effective for C^3I systems development. The panel expressed the importance of getting parallel software tool builders involved directly in C^3I testbeds and demonstrations. These should concentrate on dynamic reaction, interfaces to insert COTS fabric and COTS parallel, parallel software engineering tools and architectural independent software engineering methods.

8.1 INSERTION INTO THE C^3I ENGINEERING PROCESS

Engineers need tools at a system level scope when engineering parallel software systems. This includes the following:

- design modeling of parallel computer hardware and software in a hierarchically staged process -- focused on the tough applications that C^3I systems require
- tools for programmability-with- performance on high-time-value applications
- tools to integrate high-time-value applications with both existing C^3I applications and the future fabric described by the SNAP model
- tools to reduce the risk of achieving good results on tough character C^3I applications, focusing on the toughest types tools that predict performance and factors on any mission character

8.1.1 Integration with other subsystems

Parallel software engineering would be most valuable if more closely associated with exercises and demonstrations of C^3I systems. The recommendations were to attain tools and processes for the following:

- on-line monitoring of C^3I environment for capturing high integrity and time value
- on-line monitoring of C^3I environment for determining capabilities and resources required
- create processes for designing and implementing effective interfaces from parallel to other C^3I system components -- especially clusters of nodes that represent the reasonably expected SNAP-like COTS architectures

8.1.2 Support for dynamic adaptation

A critical capability was one that let the system rapidly and effectively change directions of processing. The system makes these rapid changes to respond to human interaction, mission change, and new phases of operation. The system can change mission activity to respond to better information. This may be one of the valuable assets of parallel computing when resources switch to meet any newly demanded processing. Dynamic adaptation based on decisions made by the C^3I system would require parallel computing for artificial intelligence and knowledge-based decision making. In addition, the parallelism and dynamic adaptation allow the system to provide fault tolerance and resiliency by determining its state and responding with a reconfiguration that meets mission needs to the best of its capability.

8.1.3 Architectural independent software representations

The industry needs an architectural independent programming model. Defining such a model has many difficulties because of the diversity of application character and architectural capability. The panel and position makers described several approaches that build on a hierarchy of layers of design, architectural selection then machine selection. In addition, automatic parallel program generators for these layers are the subject of research of several position makers. However, none had applied these methods to the tough application character type problems that are similar to C^3I ones. Air traffic control was the only difficult example given by the researchers. A problem seems to be that the architectural targets have been parallel

computers and tools built primarily for scientific research. Due to the significant difference in character and requirements, a simple transfer of this technology from scientific research might not be effective. This does not lessen the need for the hierarchical tools.

8.1.4 C^3I Parallel Tool Requirements

The panel suggested the following tool requirements:

- coupling of C^3I parallel software with application modeling
- integration analysis tools to provide interfaces with other systems & drivers
- coordination languages for design, applications and integration
- tools to find and identify the optimum boundary between COTS SNAP-like clusters and more tightly coupled parallel computers
- tools to allow the maturity of the SNAP-like COTS fabric to be evaluated to determine when to apply software enhancements to gain the C^3I factors
- methods to gain involvement in C^3I exercises, operations, activities to gain better software operational level knowledge
- cost models and prediction methods for time constrained applications, tough application character and addition of C^3I mission factors to parallel applications

The panel recommended that designers include the following testing and verification mechanisms in their designs:

- probes that allow testing parallel computers in the mix of subsystems and specialized architectures in C^3I systems
- regression testing tools for parallel executable codes during system verification

8.2 PLANNED RESEARCH FOCUS

The goal of parallel software engineering is to create development processes and architectures to meet both high-time-value goals and high-integrity-value goals within the parallel HPC environment. Applications of any character should be feasible with performance, programmability, and portability. In order to accomplish this daunting task, parallel software engineering must concentrate on the most difficult character applications. These are the ones that have extensive complexity, dynamic resource demands, symbolic interaction and frequent interaction, ie, the toughest problem types. However, parallel software engineering must recognize the need to integrate those results with existing C^3I systems and the vision of commodity SNAP-like architectures.

8.2.1 Strategic Research Areas

The general areas of research recommended are the following:

- system supervisor and control -- create the necessary additions to commodity COTS for control of the SNAP-like architecture and interfaces to high-time-value nodes
- architectural independence -- C^3I must apply to latest processor and architecture capabilities, therefore the need to continually port to new architectures is critical to maintaining military advantage
- non intrusive entry and coordination with C^3I systems to solve latest needs -- varied operational modes and demands require that information be collected to better design and refine the use of parallel computing in C^3I systems

- interface and cooperation with industry for obtaining an effective COTS a common fabric - - requires advanced emulation, including methods of obtaining C^3I system factors like reliability, fault tolerance, security, graceful degradation, and dynamic resource change
- track commercial advances in distributed processing, coordination, operating system, data or object base and language software
- adopt new technology into the C^3I system model as long as there is a benefit -- this is reasonably expected to be the least expensive manner to do a majority of the mission needs
- develop standardized methods to meet the C^3I mission factors and the tough problem needs, yet providing rapid software development and easy maintenance and enhancements

8.2.2 Short Term Research

There are several short term limitations whose study would resolve major issues. If solved they would also be effective in the commercial-decision-support system industry. Some of these study areas and limitations are the following:

- determine a suitable organization of C^3I data -- in place of SQL relational organization -- experiments were suggested by the panel to determine the limitations and advantages
- generate an application framework for creating dynamic resource changes in parallel systems -- study the capabilities required to respond
- create monitoring and testing tools to reduce uncertainty of correctness, performance, and mission factors
- evaluate operating system, interconnect access protocols, and network protocols that limit parallel performance, reaching C^3I factors, portability, reuse and performance prediction
- determine the programming models and matching parallel architectures that can resolve the time uncertainty problems of parallel computers
- find methods of achieving high-time-value performance and gaining high-integrity-value using code reuse technology
- continue to support system engineering tools that can provide the applications allocation to SNAP-like COTS systems combined with HPC multiprocessors
- develop a hierarchy of executable specification tools for high-time-value design and development layering

The executable specification tools should include the following ones:

- provide for engineering of C^3I mission factors, performance, and multiple application character needs
- create performance prediction tools -- relate to a hierarchy of design stages that would be useful to the system designer
- relate the performance prediction tools to cost models
- define the hierarchy to allow the removal of the uncertainty of availability of specific processor architectures
- provide tools that give design visibility at each level in the hierarchy
- provide reverse engineering capability for understanding of designs

These issues need focus so they can gain the proper priority, feasibility, time sequencing dependencies and budgets.

*U.S. GOVERNMENT PRINTING OFFICE: 1995-710-126-20067

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.